LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-PROC-208611

DUNE - a granular flow code

D. M. Slone, T. L. Cottom, W. B. Bateson

December 15, 2004

Nuclear Explosives Code Developers Conference
Livermore, CA, United States
October 4, 2004 through October 7, 2004

Proceedings from the NECDC 2004

DUNE – a granular flow code (U)

D.M. Slone,* T.L. Cottom,* W.B. Bateson*

*Lawrence Livermore National Laboratory, Livermore, California, 94550

DUNE was designed to accurately model the spectrum of granular. Granular flow encompasses the motions of discrete particles. The particles are macroscopic in that there is no Brownian motion. The flow can be thought of as a dispersed phase (the particles) interacting with a fluid phase (air or water). Validation of the physical models proceeds in tandem with simple experimental confirmation. The current development team is working toward the goal of building a flexible architecture where existing technologies can easily be integrated to further the capability of the simulation. We describe the DUNE architecture in some detail using physics models appropriate for an imploding liner experiment. (U)

Introduction

Granular flow encompasses the motions of discrete particles or grains. The particles are macroscopic in that there is no Brownian motion. The flow can be thought of as a dispersed phase (the grains) interacting with a fluid phase (air or water). Another way to look at granular flow is as a continuum between dilute and concentrated flows. The first extreme is when grains do not collide and granular kinetic stresses may dominate while a carrier fluid supports the grains (e.g. dust devils) – which leads to multiphase flow. The latter extreme occurs when the grains are in permanent contact with each other (e.g. the sand pile in an hourglass after time has expired) – grains roll, rub, and scrape together under frictional stresses.

DUNE was designed to accurately model the spectrum of granular flow by the time the code is mature. Validation of the physical models proceeds in tandem with simple experimental confirmation. The current development team is working toward the goal of building a flexible architecture where existing technologies can easily be integrated to further the capability of the simulation.

The DUNE Architecture

The general architecture for DUNE was planned so that new parts of the simulation process would be able to plug in through a backbone. DUNE is not a single computational code; rather it is a physical model enabler and coupler. A GUI driver, see Fig. 1, integrates the tools that initialize the simulation parameters, and the driver executes appropriate modules and codes as required by the input and run-time state of a

simulation. Expectations are that DUNE will interact with other large codes as either libraries, spawned sub-processes, or through a shared data management layer.

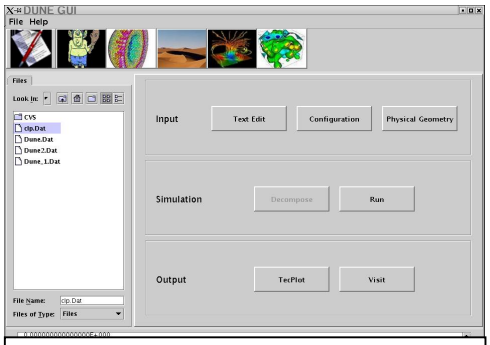


Fig. 1 - DUNE GUI

The DUNE executable is designed to be self-CASE'ing - the executable for a particular simulation will be built according to the physical models required by the specified input parameters. An example is friction models. DUNE will have one or two default models, instantiated by keyword, but the user can point to a library or executable (assuming that package subscribes to the DUNE friction model's interface), or include in the input deck a MATLAB-like description of a friction model that DUNE will translate into executable code.

Judicious code reuse and effective utilization of existing tools allowed the prototype version of DUNE to be delivered on time. Some of the tools incorporated were generally available and downloaded from the web, while others were developed by other organizations or projects at LLNL. JBuilder, provided by Borland™, was used to develop the GUI driver application. Subsequent modifications to the GUI's look-and-feel by the development team typically took less than an hour of effort. Cyclops, an XML driven JAVA application, was incorporated to drive the input parameter specification for DUNE. Cyclops was developed in B-Division to handle input specifications for a variety of other physics based codes. Integrating the DUNE syntax into the Cyclops framework was easy since Cyclops creates its windowing structure based solely on XML descriptions of the input data. The geometry definition phase of the model uses the Overture-Rapsodi software package from CASC. The two teams worked together to provide a liner description to the available solid models and incorporate the particle-filling algorithm.

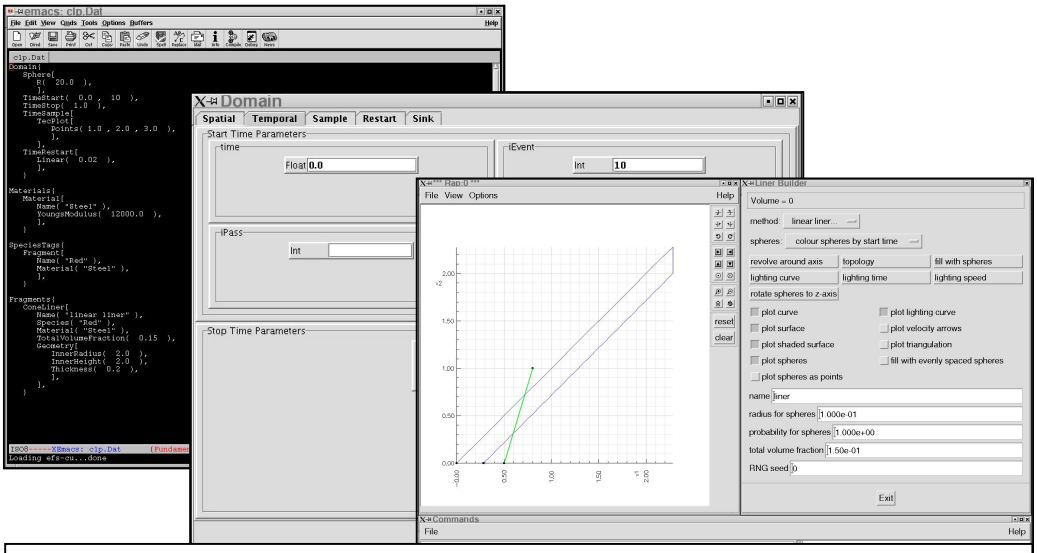


Fig. 2 – Multiple views of the DUNE simulation (input) file. From left: text view, context aware view, geometry instantiation view.

The development team is hoping to leverage Vista, a data management tool developed in B-Division, to ease the task of data sharing in the data object layer of DUNE.

An adaptable architecture requires a highly flexible yet manageable input specification. DUNE was designed with a hierarchical input notation from the beginning. One benefit is that, with just a couple of minor tweaks, the input syntax lent itself nicely to XML. An XML representation of the input file makes interacting with specific scopes of the input easier. The input was developed such that DUNE uses a single file for initialization and restarts. The code selectively updates the input deck to reflect the current state of the simulation. For example, the initial input may include a liner geometry specification in terms of the encompassing shape. After the geometry is appropriately filled with particles, this new representation is added to the input and the liner geometry specification transforms to a comment. Such representations allow for multiple, concurrent views for the simulation file. One realization might include, for example, a text view (xemacs), a context aware view (Cyclops), and a geometry instantiation view (Overture), see Fig. 2. Updating the text view leads to the context aware to update itself. Updating the geometry instantiation view leads to both the text and context aware view to update appropriately.

The visualization of the simulation is handled by the commercial product Tecplot or the B Division code VisIt. For VisIt, we developed a plug-in to visualize the particles and have let the VisIt group maintain the plug-in with their standard distribution – freeing the DUNE developers from having to ensure that the plug-in is kept current with the latest VisIt release on every platform.

Physics modeling in DUNE

DUNE was designed with the notion of plugging in various physics modules for differing physical objects. Each physical object is wrapped in a contact mechanics aware membrane. Differing objects communicate through their membrane. Expectations are that DUNE will interact with FEM codes to generate accurate particle size and location distributions based on failure models. Depending on stresses, the particles will retain evolution of their volume or become rigid objects. Hydrodynamic codes may be used to model the interactions of the particles with ambient air and obstacles. The data object layer will ease and encapsulate the task of coupling DUNE to interface with such codes. We only need to agree on the data each code needs, what data each code evolves, and communicate the appropriate queries to the database. The GUI will run other modules in whichever fashion they deem acceptable – DUNE won't enforce a particular notion of code execution upon existing codes.

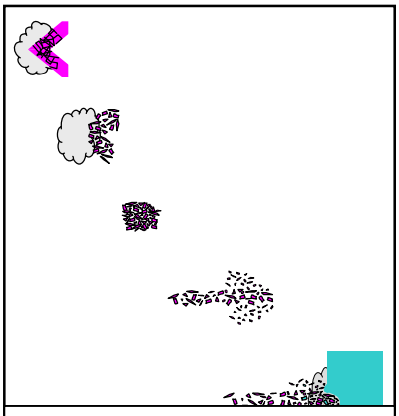


Fig. 3 – Five stages in the imploding liner. Time proceeds from top to bottom.

Stages in the model

There are five stages, see Fig. 3, in the end-to-end model for the shape charge. While some of the physics phenomena are applicable across stages,

their scale length may not be the same. The initial stage is the **Driven Liner**, governed by hydrodynamics, material failure, and granular flow. Next is the **Lift Off of Fractured Liner**, which requires both granular flow and hydrodynamics models. The **Convergence** stage follows, governed by granular flow, fracturing, and hydrodynamics. The penultimate state is that of **Streaming To Target**, again driven by granular flow and hydrodynamics models. **Target Interaction** is the final stage, needing granular flow, hydrodynamics, material failure and chipping, and material fracture models. It is likely that some stages will involve models not listed here as the simulations mature.

Contact mechanics

In order to impart more fidelity to the granular flow models, we will account for the finite transit time that particle contact mechanics require. This is referred to as causality. The geometric surface of a particle moves when the impulse from the contact arrives, see Fig 4. The center of mass moves from the force at some transient time dependent upon the material properties. Appropriately applying this type of model requires that the contact time be short relative to the transit time.

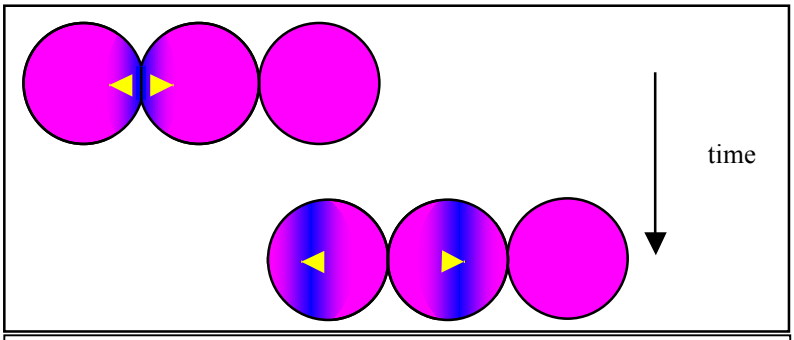


Fig. 4 – Modeling the finite transit time for a contact to propagate

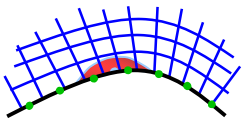


Fig. 5 – Membrane (green dots) for the object (blue mesh) defined at the surface (black line).

Coupling objects with surfaces

One of the most interesting aspects of the DUNE data architecture is the notion that everything is an abstract object. That is, each solid (such as a particle or a mesh) or liquid (such as an ambient background material) is an entity unto itself. Each object transmits information about its state through contact mechanics. This is done by wrapping each object with a contact mechanics membrane.

A membrane is represented by an analytic function defined by points on the surface of the object, see Fig 5. This interpretation of the object means that each surface sees other surfaces as deformable objects with a center-of-mass drift and rotation. This allows the hydrodynamics within objects to be isolated from each other. Object information is transferred across surface membranes into their corresponding volumes. Properties such as tension, friction, contact

mechanics including impulse information, mass, momentum, energy fluxes, and wetting are typical of object state communicated across membranes.

Experimental Validation

Molecular dynamics-like experiments

We have begun an experimental campaign to validate the granular flow physics. There are experiments targeted at particle interactions, in a molecular dynamics sense, with increasing complexity. The simplest is a ball bouncing up and down on a plate and picking up rotation. Other experiments include two and three balls colliding including rotation, bouncing a column of balls on a plate with varying separations, a single ball slipping on a surface and picking up rotation, and two balls slipping and colliding on a surface, see Fig 6. This set of experiments leads up to the most complex one - a set of converging balls colliding on a plate. These experiments are designed to isolate a particular effect (such as imparting rotation to slipping or colliding particles) so that the implementation in DUNE is validated in a stepwise fashion before proceeding to more integrated simulations.

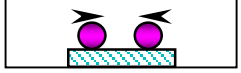


Fig. 6 - Rolling and slipping balls collide

Entrainment experiments

In a similar vein, a series of experiments is planned to validate the particle entrainment physics. The experiments will look at single balls or ellipses in a liquid flow, pairs, and then multiple particles in the liquid flow, see Fig 7. A subsequent series will repeat the effort with just balls, but using a mach 3-7 gas flow.

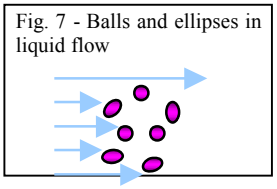


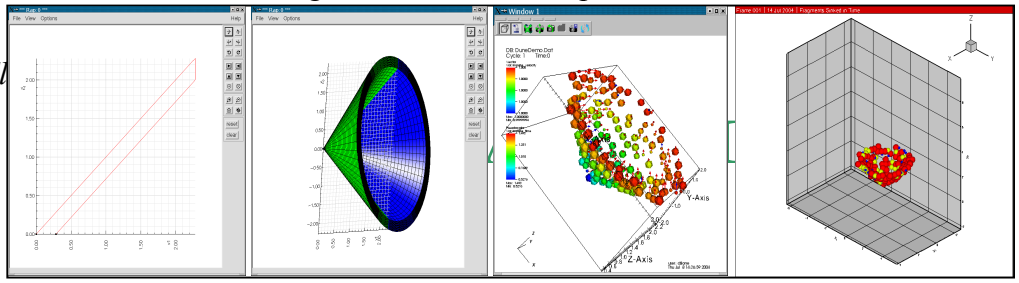
Fig. 7 - Balls and ellipses in liquid flow

Current progress

The DUNE team has delivered a prototype code that demonstrates features of both data architecture and granular flow physics. Users can define a cone liner; specify the volume fraction of the liner fill and discrete radii for the spheres in the fill. Overture is used to convert the liner description into an instantiation of granular particles filling the liner volume. The resulting particles are given an initial velocity normal to the cone surface with a magnitude dependent on their distance from the cone axis. Each particle has its own impulse dependent on its distance from the apex of the liner. The particles then evolve with inelastic collisions using force laws until the user specified stopping time. Either VisIt or Tecplot can be used to display the simulation.

We are currently implementing the multiple, concurrent views of the simulation file. At present the views are seen sequentially. We are actively investigating the Vista and BABEL tools for their applicability as the DUNE data layer. The particle loading is being improved with faster fill algorithms and more particle distributions. Also, the initial

Sl



friction model is being implemented, along with particle rotation. We have begun looking at issues associated with running large numbers of particles on parallel machines.

Fig. 8 – Illustration of prototype capability. From left: describe liner profile, generate 3D liner, fill liner with spheres of given radii with corresponding initial velocities and impulses, visualize the time history of the particle trajectories.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.